/\*selecteer alle familienaam uit actor en voeg daarna een tweede kolom toe die het aantal per naam zal weergeven. Sorteer dan volgens het aantal van hoog naar laag \*/

SELECT actor.last\_name, COUNT(actor.last\_name) as aantal

FROM actor

GROUP BY actor.last\_name

ORDER BY aantal DESC;

1.

SELECT DISTINCT\*

FROM country;

* Zo weet je hoeveel verschillende er effectief zijn

SELECT COUNT(country.country)

FROM country;

2.

SELECT country.country, COUNT(customer.customer\_id) AS aantal

FROM country LEFT JOIN city ON country.country\_id = city.country\_id

LEFT JOIN address ON city.city\_id = address.city\_id

LEFT JOIN customer ON address.address\_id = customer.address\_id

GROUP BY country.country

ORDER BY aantal DESC

LIMIT 5;

3.

SELECT language.name

FROM language

ORDER BY language.name ASC;

4.

SELECT actor.first\_name,actor.last\_name

FROM actor

WHERE actor.last\_name LIKE '%son%'

ORDER BY actor.first\_name ASC;

5.

/\*veel op veel relatie\*/

SELECT film.title,category.name

FROM film LEFT JOIN film\_category ON film.film\_id = film\_category.film\_id

LEFT JOIN category ON film\_category.category\_id = category.category\_id;

6.

SELECT category.name, COUNT(film.title)

FROM category LEFT JOIN film\_category ON category.category\_id = film\_category.category\_id

LEFT JOIN film ON film\_category.film\_id = film.film\_id

GROUP BY category.name;

7.

SELECT actor.first\_name, actor.last\_name, COUNT(film.title)

FROM actor LEFT JOIN film\_actor ON actor.actor\_id = film\_actor.actor\_id

LEFT JOIN film ON film\_actor.film\_id = film.film\_id

GROUP BY actor.first\_name, actor.last\_name;

8.

SELECT film.film\_id,film.title, inventory.film\_id

FROM film LEFT JOIN inventory ON film.film\_id = inventory.film\_id

WHERE inventory.film\_id IS NULL;

* Laatste kolom moet niet getoond worden

SELECT film.film\_id,film.title

FROM film LEFT JOIN inventory ON film.film\_id = inventory.film\_id

WHERE inventory.film\_id IS NULL;

9.

SELECT customer.first\_name, customer.last\_name, customer.customer\_id, rental.customer\_id

FROM customer LEFT JOIN rental ON customer.customer\_id = rental.customer\_id

GROUP BY customer.first\_name, customer.last\_name;

* Onnodige kolommen verbergen

SELECT customer.first\_name, customer.last\_name

FROM customer LEFT JOIN rental ON customer.customer\_id = rental.customer\_id

WHERE rental.rental\_date IS NULL

GROUP BY customer.first\_name, customer.last\_name;

10.

SELECT COUNT(film.title), category.name, language.name

FROM category

LEFT JOIN film\_category ON category.category\_id = film\_category.category\_id

LEFT JOIN film ON film\_category.film\_id = film.film\_id

LEFT JOIN language ON film.language\_id = language.language\_id

WHERE category.name ='Documentary' AND language.name='English'

GROUP BY category.name, language.name;

11.

SELECT\*

FROM actor

WHERE first\_name = 'SCARLETT';

12.

SELECT\*

FROM actor

WHERE last\_name = 'Johansson';

13.

SELECT COUNT(DISTINCT actor.last\_name)

FROM actor;

14. /\* Welke namen worden niet herhaald ?\*/

SELECT actor.last\_name, COUNT(actor.last\_name) as aantal

FROM actor

GROUP BY actor.last\_name

HAVING aantal = 1;

15.

SELECT actor.last\_name, COUNT(actor.last\_name) as aantal

FROM actor

GROUP BY actor.last\_name

HAVING aantal > 1;

OF

SELECT actor.last\_name

FROM actor

GROUP BY actor.last\_name

HAVING COUNT(actor.last\_name) > 1;

* Hier zie je geen kolom met aantallen

16.

SELECT actor.actor\_id, actor.first\_name, actor.last\_name, COUNT(actor.actor\_id) AS aantal

FROM actor LEFT JOIN film\_actor ON actor.actor\_id = film\_actor.actor\_id

LEFT JOIN film ON film\_actor.film\_id = film.film\_id

GROUP BY actor.actor\_id, actor.first\_name, actor.last\_name

ORDER BY aantal DESC

LIMIT 1;

SELECT actor.actor\_id, actor.first\_name, actor.last\_name

FROM actor LEFT JOIN film\_actor ON actor.actor\_id = film\_actor.actor\_id

LEFT JOIN film ON film\_actor.film\_id = film.film\_id

GROUP BY actor.actor\_id, actor.first\_name, actor.last\_name

HAVING COUNT(actor.actor\_id)

ORDER BY COUNT(actor.actor\_id) DESC

LIMIT 1;

17.

SELECT film.title, inventory.store\_id, rental.rental\_date, rental.return\_date

FROM rental LEFT JOIN inventory ON rental.inventory\_id = inventory.inventory\_id

LEFT JOIN film ON inventory.film\_id = film.film\_id

WHERE film.title='Academy Dinosaur';

* Huidig rental aanbod

SELECT film.title, inventory.store\_id, rental.rental\_date, rental.return\_date, inventory.inventory\_id

FROM rental LEFT JOIN inventory ON rental.inventory\_id = inventory.inventory\_id

LEFT JOIN film ON inventory.film\_id = film.film\_id

WHERE film.title='Academy Dinosaur' AND inventory.store\_id='1' AND rental.return\_date IS NOT NULL;

OF

SELECT film.title, inventory.store\_id,inventory.inventory\_id, count(inventory.inventory\_id) as aantal

FROM rental LEFT JOIN inventory ON rental.inventory\_id = inventory.inventory\_id

LEFT JOIN film ON inventory.film\_id = film.film\_id

WHERE film.title='Academy Dinosaur' AND inventory.store\_id='1' AND rental.return\_date IS NOT NULL

GROUP BY inventory.inventory\_id;

18. /\*voeg een record in om Mary Smith te vertegenwoordigen bij het huren van ‘Academy Dinosaur’ van Mike Hillyer in Store 1 vandaag.\*/

19.

SELECT film.title, inventory.store\_id, rental.rental\_date, rental.return\_date, inventory.inventory\_id, film.rental\_duration, ADDDATE(rental.rental\_date,film.rental\_duration) AS due\_date

FROM rental LEFT JOIN inventory ON rental.inventory\_id = inventory.inventory\_id

LEFT JOIN film ON inventory.film\_id = film.film\_id

WHERE film.title='Academy Dinosaur' AND rental.return\_date IS NULL;

—> overbodige kolommen verdwijnen

SELECT rental.rental\_date, ADDDATE(rental.rental\_date,film.rental\_duration) AS due\_date

FROM rental LEFT JOIN inventory ON rental.inventory\_id = inventory.inventory\_id

LEFT JOIN film ON inventory.film\_id = film.film\_id

WHERE film.title='Academy Dinosaur' AND rental.return\_date IS NULL;

Typeoplossing:

SELECT rental.rental\_date, rental.rental\_date + INTERVAL(SELECT film.rental\_duration FROM film WHERE film.film\_id = 1) day AS due\_date

FROM rental

WHERE rental.rental\_id = (SELECT rental.rental\_id FROM rental ORDER BY rental.rental\_id DESC LIMIT 1);

20. /\* Wat is de gemiddelde looptijd (duur) van alle films in de sakila DB?\*/

SELECT AVG(film.length)

FROM film;

21. /\* Wat is de gemiddelde looptijd (=duur) van films per categorie\*/

SELECT category.name, AVG(film.length)

FROM film

LEFT JOIN film\_category ON film.film\_id = film\_category.film\_id

LEFT JOIN category ON film\_category.category\_id = category.category\_id

GROUP BY category.name

ORDER BY AVG(film.length) DESC;

Type oplossing:

SELECT category.name, AVG(film.length)

FROM film

JOIN film\_category USING(film\_id) join category USING(category\_id)

GROUP BY category.name

ORDER BY AVG(film.length) DESC;

22. /\* Toon de voor- en achternaam van alle acteurs van de tabel acteur\*/

SELECT actor.first\_name, actor.last\_name

FROM actor;

23. /\* Toon de voor- en achternaam van elke acteur in één kolom in hoofdletters. Noem de kolom acteurnaam \*/

SELECT CONCAT(UPPER(actor.first\_name)," ", UPPER(actor.last\_name)) AS acteurnaam

FROM actor;

24. /\* Je moet het ID-nummer, de voornaam en de achternaam van een acteur vinden, van wie je alleen de voornaam kent, ‘Joe’. Schrijf de select hiervoor. \*/

SELECT actor.actor\_id, actor.first\_name, actor.last\_name

FROM actor

WHERE actor.first\_name = 'Joe';

25. /\* Zoek alle acteurs wiens achternaam de letten GEN bevat \*/

SELECT \*

FROM actor

WHERE actor.last\_name LIKE '%GEN%';

26. /\* Zoek alle acteurs wiens achternaam de letters LI bevat. Bestel deze keer de rijen op achternaam en voornaam, in die volgorde \*/

SELECT actor.first\_name, actor.last\_name

FROM actor

WHERE actor.last\_name LIKE '%LI%'

ORDER BY last\_name, first\_name ASC;

27. /\* Gebruik IN om de land\_id en landkolommen van de volgende landen weer te geven: Afghanistan, Bangladesh en China \*/

28. /\* Hoeveel verschillende unieke namen zijn er (actors) \*/

SELECT COUNT(DISTINCT actor.last\_name)

FROM actor;

29. /\* Welke namen worden niet herhaald? \*/

SELECT actor.last\_name

FROM actor

GROUP BY actor.last\_name

HAVING COUNT(actor.last\_name) = 1;

30. /\* Welke namen verschijnen meer dan eens \*/

SELECT actor.last\_name

FROM actor

GROUP BY actor.last\_name

HAVING COUNT(actor.last\_name) > 1;

31. /\* Welke acteur is in de meeste films verschenen? \*/

SELECT actor.actor\_id, actor.first\_name, actor.last\_name

FROM actor

LEFT JOIN film\_actor ON actor.actor\_id = film\_actor.actor\_id

LEFT JOIN film ON film\_actor.film\_id = film.film\_id

GROUP BY actor.actor\_id, actor.first\_name, actor.last\_name

HAVING COUNT(film.film\_id)

ORDER BY COUNT(film.film\_id) DESC

LIMIT 1;

32. /\*Selecteer de meest gehuurde films omgekeerd gesorteerd \*/

SELECT film.film\_id, film.title, COUNT(rental.inventory\_id)

FROM film

LEFT JOIN inventory ON film.film\_id = inventory.film\_id

LEFT JOIN rental ON inventory.inventory\_id = rental.inventory\_id

GROUP BY film.film\_id, film.title

ORDER BY COUNT(rental.inventory\_id) DESC;

33. /\*Hoeveel exemplaren van de film Hunchback Impossible bestaan er?\*/

34.

SELECT staff.first\_name, payment.amount,payment.payment\_date

FROM staff

LEFT JOIN payment ON staff.staff\_id = payment.staff\_id

WHERE payment.payment\_date LIKE '2005-08%';

OF

SELECT CONCAT(staff.first\_name, ' ', staff.last\_name) as naam, SUM(payment.amount) AS som

FROM staff

LEFT JOIN payment ON staff.staff\_id = payment.staff\_id

WHERE payment.payment\_date BETWEEN '2005-08-01' AND '2005-08-31'

GROUP BY naam;

35. /\*Lijst het totaal op die door de klant betaald werd, alfabetisch gesorteerd\*/

SELECT customer.customer\_id, customer.last\_name, customer.first\_name, SUM(payment.amount) AS totaal

FROM customer

LEFT JOIN payment ON customer.customer\_id = payment.customer\_id

GROUP BY customer.customer\_id, customer.first\_name, customer.last\_name

ORDER BY customer.last\_name, customer.first\_name ASC;

36. /\* Selecteer title, beschrijving, rating, film lengte en toon alle films van langer dan 3uur\*/

SELECT film.title, film.description, film.rating, film.length

FROM film

WHERE film.length >=180;

37/\*Selecteer het betalings id, bedrag en betaaldatum uitgevoerd na 27/5/2005 \*/

SELECT \* /\*payment.payment\_id, payment.amount, payment.payment\_date\*/

FROM payment

WHERE payment.payment\_date > '2005-05-27'

ORDER BY payment.payment\_date ASC;

38. /\*Selecteer alle klanten waarvan de familienamen beginnen met een s en de voornamen eindigen met een n\*/

SELECT customer.customer\_id, customer.last\_name, customer.first\_name

FROM customer

WHERE customer.last\_name LIKE 's%' AND customer.first\_name LIKE '%n';

39. /\*Selecteer alle niet-actieve klanten waarvan de familienaam begint met een m \*/

SELECT \*

FROM customer

WHERE customer.active = 0 AND customer.last\_name LIKE 'm%';

40. /\*Selecteer alle kolommen van de category tabel waar de primary key groter is dan 4 en de naam begint met C of S of T\*

SELECT \*

FROM category

WHERE category.category\_id > 4 AND category.name LIKE 'c%' OR category.name LIKE 's%' OR category.name LIKE 't%';

41. /\*Selecteer alle kolommen zonder de kolom met het paswoord van de staff tabel voor rijen die een paswoord bevatten\*/

SELECT staff.staff\_id, staff.first\_name, staff.last\_name, staff.address\_id, staff.picture, staff.email, staff.store\_id, staff.active, staff.username, staff.last\_update

FROM staff

WHERE staff.password IS NOT NULL;

42. /\*Toon alle namen van acteurs die eenzelfde familienaam delen door ten minste 2 acteurs\*/

SELECT actor.last\_name, COUNT(actor.last\_name) as aantal

FROM actor

GROUP BY actor.last\_name

HAVING aantal >= 2;

43. WACHTEN